Automatic Overset Grid Generation with Heuristic Feedback Control

Peter I. Robinson
QSS Group
NASA Ames Research Center
Moffett Field CA 94035
robinson@ptolemy.arc.nasa.gov

November 5, 2001

Abstract

An advancing front grid generation system for structured Overset grids is presented. The grid generation system automatically modifies Overset structured surface grids and control lines until user-specified grid qualities are achieved. The system is demonstrated on two examples. The first example refines a space shuttle fuselage control line until global truncation error is achieved. The second example advances, from control lines, the space shuttle orbiter fuselage top and fuselage side surface grids until proper overlap is achieved. The system generates surface grids in minutes for complex geometries.

The grid generation system is implemented as a heuristic feedback control (HFC) expert system which iteratively modifies the input specifications for Overset control line and surface grids. It is developed as an extension of modern control theory, production rules systems and subsumption architectures. The methodology provides benefits over the full knowledge lifecycle of an expert system for knowledge acquisition, knowledge representation and knowledge execution. The vector/matrix framework of modern control theory is used to systematically acquire and represent expert system knowledge. Missing matrix elements imply missing expert knowledge. The execution of the expert system knowledge is performed through symbolic execution of the matrix algebra equations of modern control theory. The dot product operation of matrix algebra is generalized for heuristic symbolic terms. Constant time execution is guaranteed.

A compilation of the grid heuristics for the Overset domain as well as a users manual are provided.

Contents

1 Executive Summary

This paper presents a closed-loop control, expert system application in the domain of automatic structured Overset surface grid generation. Automation of the surface grid generation process will significantly reduce the time and cost required to develop surface grids for aerospace vehicles (Fig. 1). In addition, a theory of heuristic feedback control (HFC) for developing and delivering closed-loop control expert systems is also presented. The primary benefit of the theory will be to reduce the time and cost many expert system developers experience addressing the knowledge acqusition bottleneck. Additional knowledge representation and execution benefits are also presented.

The grid generation system provided the first applications of automatically controlling, through heuristic feedback control methods, Overset grid tools SRAP and SURGRD. The system is demonstrated on two representative grid generation problems. The first problem shows the results of the grid controller iteratively modifying a control line (Fig. 4). The second problem shows the results of the grid controller marching two space shuttle surface grids together until proper overlap is achieved (Fig. 5). The performance of the grid controller marching two space shuttle surface grids can be seen in Fig. 6. The automatic grid generation software system was developed by formalizing over 30 domain heuristics. The domain heuristics for the grid controller are ennumerated in Appendix A.

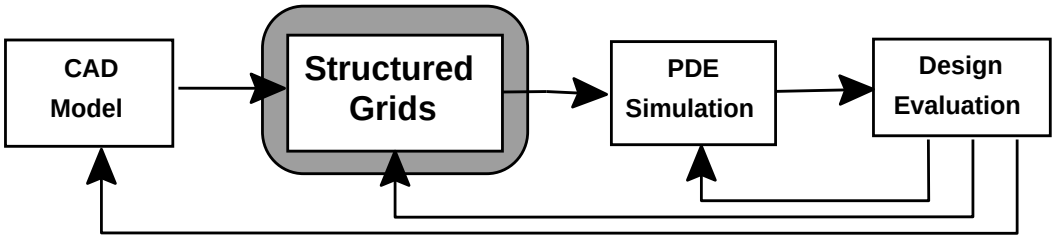


Figure 1 Conceptual design of an aerospace vehicle is a four step process. CAD models are constructed from requirements. Structured grids are generated about the CAD models. Simulations are run over the grids. Simulation results are used to evaluate the conceptual design. Reduction in the time and cost for developing Overset structured surface grids is accomplished through automation.

The execution of the grid controller system follows a feedback control cycle (Fig. 2). The algorithm is based on the insight that the feedback process which grid designers utilize to design grids is an instance of a feedback control system (Fig. 3). Given an input file from the user (Fig. 7), on each iteration, the grids are sensed (Fig. 8). Automatic grid quality routines SGQ[1] and OLGQ[2] derive grid quality state variables (Fig. 9, left) from grid sensors. The grid qualities are matched against grid heuristics (Fig. 10). The matched grid heuristics determine the next grid actions (Fig. 9, right). The grid actions are automatically compiled into input specification files for the control line and surface grid generation tools. These tools are executed to generate new control lines and surface grids. The process repeats until the user defined surface grid and control line objectives are achieved. The grid controller algorithm which encodes the grid heuristics is seen in Fig. 11.

This paper demonstrates that the feedback control machinery of modern control theory [5, 38] can be generalized for heuristic systems. The benefits are realized throughout the knowledge life-cycle of the expert system for knowedge acquisition, representation amd execution. The knowledge acquisition framework helps expert system developers extract domain knowl-

[1] Goetz Klopfer, Jeff Onufer
[2] Jeff Onufer, Goetz Klopfer, Peter Robinson

edge in a systematic manner [39] (Section 7). The knowledge representation framework helps expert system developers represent and organize the heuristic knowledge using rules imbedded in matrices. Knowledge execution of the grid control laws is accomplished through matrix algebra operations. The dot product operation required to perform matrix multiplication on numeric terms is generalized from a *literal sum* of product terms to an *abstract combination* of rule applications. Constant time knowledge execution guarantees can be made due to the deterministic nature of matrix dot product operations.

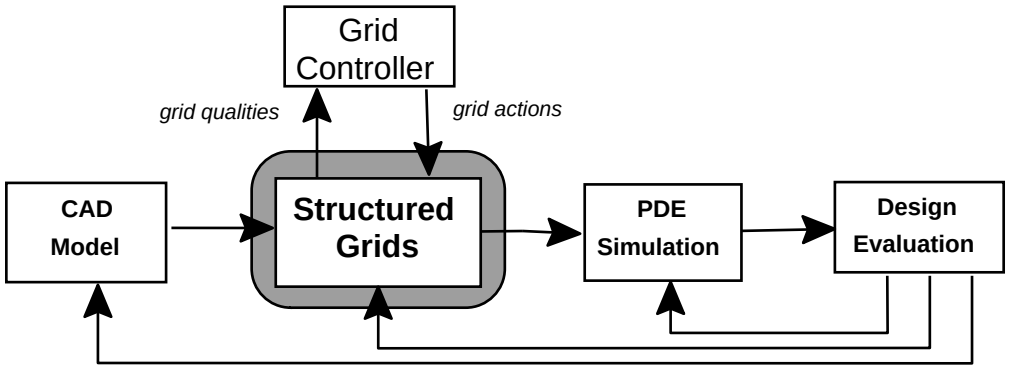


Figure 2 The grid controller modifies structured grids *before* the simulation process in order to reduce the number of mathematical "artifacts" introduced into simulation caused by discontinuities and other grid conditions. A reduction in the number of mathematical "artifacts", by ensuring that grids meet grid quality setpoints, will reduce the number of iterations required to evaluate an aerospace vehicle design.

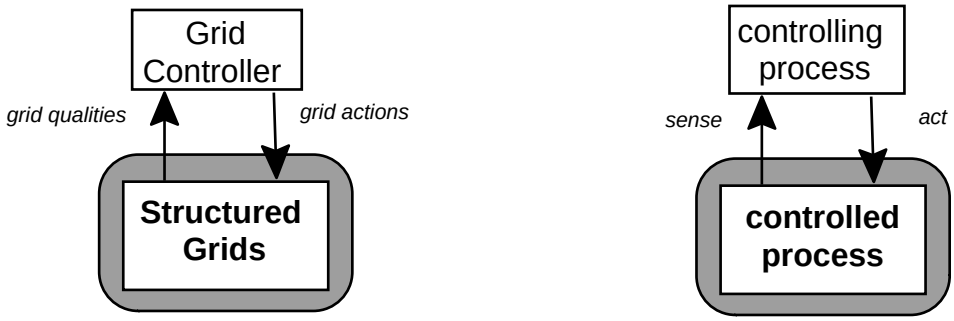


Figure 3 The grid controller is an example of a generalized feedback controller. The grid controller is *situated* in an environment of structured grids. Sensing is analogous to determining grid qualities, while acting is analogous to modifing grid parameters.

2 Motivation

There are two motivations for this work. First, to reduce the time and costs for designing aerospace vehicles, and second to reduce the time and costs for developing closed-loop expert

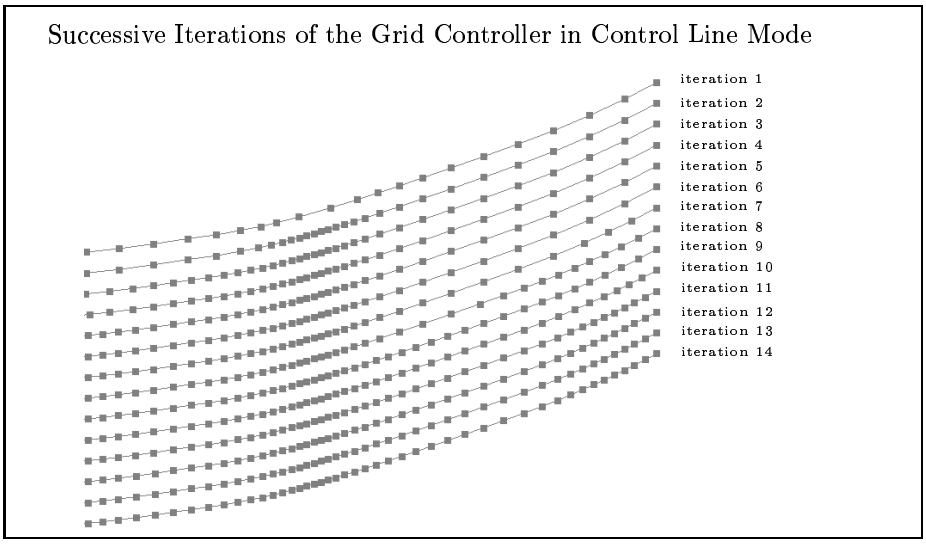


Figure 4 The grid controller iteratively modifies the fuselage side control line of the space shuttle orbiter, by iteratively generating an SRAP input specification which, when executed by SRAP, modifies the control line. Modifications include increasing the number of points as well as changing the spacing in order to achieve a user specified maximum truncation error of .004.

systems.

Reductions in time and cost for aerospace design are accomplished through the automation of the surface grid generation component of the design process. This requires grid generators which rapidly and automatically generate grids about the CAD geometry and flow features of the aerospace design. It also requires grids which support efficient and accurate simulation results. For complex griding tasks no single grid tool [28] will achieve both. Unstructured grid generators meet the first property but since unstructured grids do not follow the CAD and flow field geometry, solutions may not be efficient and accurate. On the other hand, structured grid generators generate grids which meet the second property but have significant challenges fully automating the surface grid generation process. The challenge is to develop a hybrid closed-loop expert system controller which meets both properties.

Reductions in time and cost for developing closed-loop expert systems are accomplished by the development of a methodology to reduce the knowledge acquisition bottleneck many automation projects experience. This requires methods which can symbolically reason over the axioms and propositions in a domain as well as methods which can guarantee systematic coverage of the domain knowledge. Methods from artificial intelligence provide inference procedures and qualitative reasoning methods for symbolic reasoning over complex domains. Methods from modern control theory provide quantitative vector/matrix tools to develop closed-loop process controllers and ensure systematic coverage of domain knowledge. The challenge is to develop a hybrid grid generation system which integrates the benefits of AI and conventional modelling methodologies.

3 Grid Controller Overview

The grid generation system automatically generates Overset structured grids according to a schedule of ordered tasks provided by the user as input to the grid controller (Fig. 7). Each task specifies four classes of information, namely: grid setpoints, type of control, number of

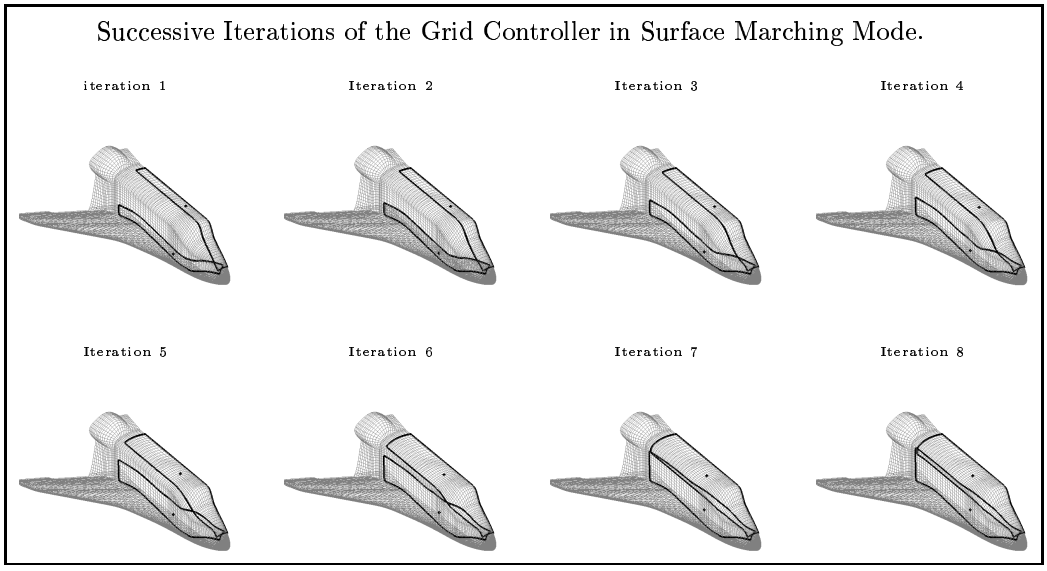


Figure 5 Two Overset structured grids corresponding to the fuselage top and fuselage side control lines are grown from control lines on a CAD model of the shuttle orbiter. The grids were grown through an iterative feedback process until overlap is achieved.

controller iterations and the initial grid states (see Appendix C for details). Tasks allow the user to decompose a grid generation problem into logical components. For example, a user may grow a subset of grids which cover smooth regions of the CAD model as its first task in the grid schedule. Once these grids are defined, they provide overlap foundation for generating grids around CAD discontinuities with additional tasks.

At each iteration, the grid controller senses the grids to determine the state variables or grid qualities (Fig. 9, left). Given grid setpoints, it computes the differences between the grid state variables and the grid setpoints. The differences drive grid control laws (Fig. 10) to determine the next grid actions (Fig. 9, right). The actions are converted into grid input specification files to be automatically executed by Overset grid generators SRAP and SURGRD. This process continues until the grid setpoints are achieved, local optima are achieved or a fixed number of iterations are completed.

The grid observations are shown in Fig. 8. The set of points and lines which connect the points define the primitive data structure of the grids. Grid generation tools modify the number of points, the position of points and the lines between points. Certain configurations of grid observations are called grid qualities. These visual predicates provide aggregate measures of the grids. Eight different grid qualities (Fig. 9, left) are utilized by the grid controller. They are divided into two categories: unary grid measures and binary grid measures. The unary grid measures are stretching ratio, truncation error, jacobians, coverage and controllability. The binary grid measures are overlap, relative volume and cell difference. A more detailed explanation of the grid qualities can be found in Appendix A. The grid controller can take seven different classes of grid actions (Fig. 9, right) which are divided into two categories: surface grid actions and control line actions. The surface grid actions modify line lengths, the number of lines, the line spacing, and grid side boundary conditions. The control line actions modify the number of control line points and point spacing.

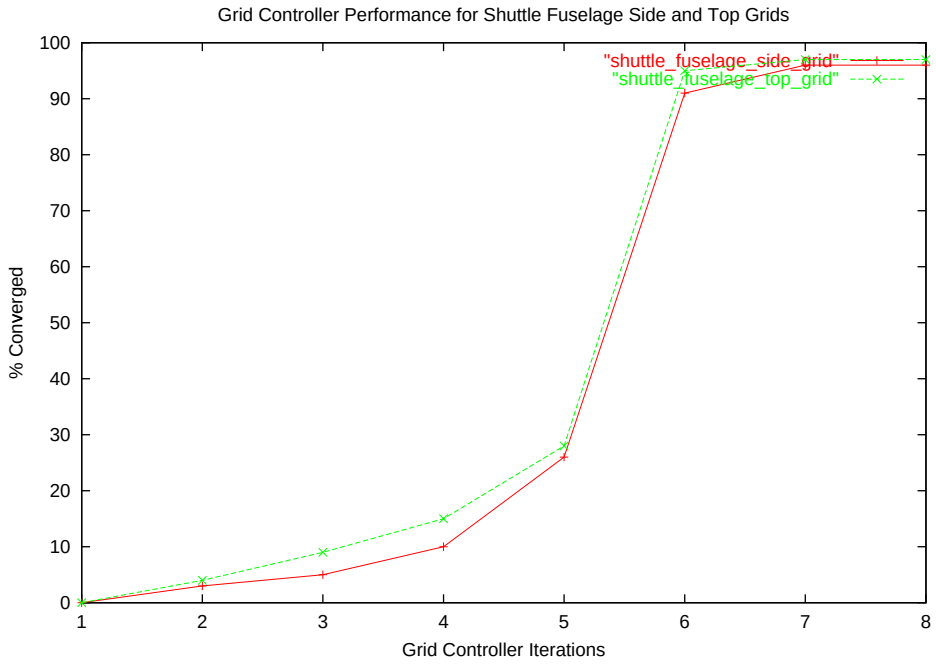


Figure 6 Measurement of grid controller performance for the shuttle fuselage side and top surface grids. Since the control lines of these two grids are approximately parallel, covergence of farfield overlap occurs on the same iteration.

4 Related Work: Grid Generation

Given a CAD reference surface of the aerospace design, the surface grid generation task for structured Overset grids can be divided into decomposition and generation tasks. The *decomposition* task is to decompose the reference surface into regions of well-behaved surfaces. It is accomplished by defining control lines on the reference surface. The *generation* task is to cover the reference surface. It is accomplished by marching the grids to and from the control lines. If the control lines represent a sufficient decomposition, the reference surface will be covered with surface grids which meet user criteria. This paper addresses the *generation* task.

4.1 Grid Controller

The grid controller is an Overset [2] structured, advancing front, grid generation system that provides the benefits of automatic surface grid generation from unstructured grid methods with the benefits of efficient and accurate simulation results of structured grid methods. A structured advancing front from a control line progressively increments the line lengths of the structured grid until overlap and other single and binary grid qualities are achieved. It is analogous to the unstructured advancing front approach, where the grid generator starts from the boundaries of grids and incrementally adds in triangles until the surface is covered [28].

Grid users at NASA Ames Research Center[3] have extended the heuristics of the grid controller and scaled its applicability to automatically generate overlapping Overset surface grids for the space shuttle orbiter model with seventeen grids, as well as the V22 Tiltrotor with

[3]Donovan Mathias and Jeff Onufer, NASA Ames Research Center

Grid Controller Schedule

Input	Description
1	Task: modify control lines
1, 0	grid 1 active, grid 2 inactive
30, 4	iterations, control line mode
0.005	truncation error setpoint
3	use user-specified control lines
2	Task: modify surface grids
1, 1	grid 1 active, grid 2 active
20, 1	iterations, surface marching mode
2	overlap setpoint
4	use current control lines

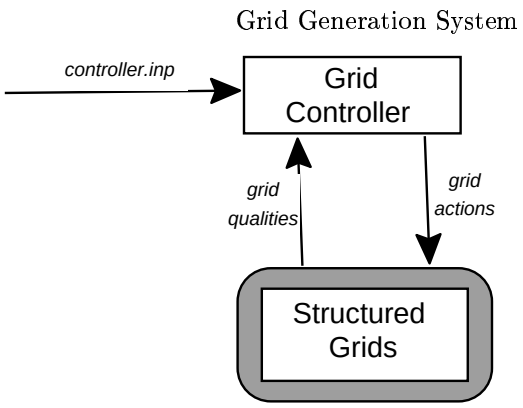


Figure 7 A grid controller schedule (left) is made up of a sequence of temporally ordered tasks. Each task is defined as a specification to the grid controller of: 1) the grid quality setpoints desired by the user, 2) the current grids, 3) the number of grid controller iterations, and 4) grid initialization methods. The grid controller schedule is the specification file for the grid controller. See Appendix C for a users manual.

fifteen grids. The validation of the automatically generated grids was achieved by running PDE simulations and deriving realistic flow solutions from the automatically generated grids. In addition, the grid controller has been integrated with a global optimization framework from Rutgers University[4]. Preliminary results from this collaboration have helped refine the grid heuristics.

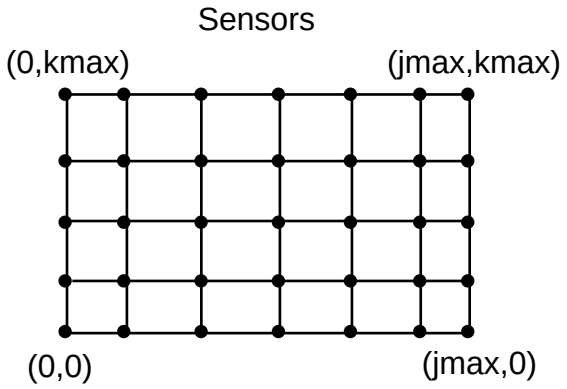
4.2 Overset Grid Generation Systems

In the Overset grid community, there has been a natural progression of increasing automation for the generation of grids. Initially grid generation tools were called directly through a command line interface. The Overgrid GUI [8] was developed to allow for rapid input parameter specification for grid generation tools via GUI templates. It also provides means to execute Overset grid tools and visualize their results.

An Overset script system [40, 35] has been developed which can assist in the CFD analysis of high-lift aerospace configurations which contain fuselage, wing, leading and trailing edge devices, engine, nacelle, strut/pylon and tail components. The script system assumes that the surface grids are defined. It has benefits of generating "error-free input files" [40] through reliance on a single repository of common information related to the aerospace design. The grid controller should be another routine invoked by this script system to generate the surface grids and proper overlap required by the scripting system. The user defined grid setpoints, such as overlap and truncation error used by the grid controller, should be set by the script system due to the heuristics in the script system based on characteristic lengths from CAD geometry and flow field resolution heuristics from simulation scenario (Reynolds number and angle of attack) requirements. The grid controller is similar to the script system in that it maintains a repository of grid knowledge from which error-free input files for SURGRD and SRAP are generated. The grid controller differs from the script system in that it does not assume the CAD geometry is made up of a predefined set of aircraft components.

PEGASUS [45] cuts portions of Overset surface grids away to ensure proper overlap between grids through a process called hole-cutting. This approach produces variable kmax grids which

[4]Dr. Don Smith, Dr. Khaled Rasheed, and Yan Meng, Rutgers University



$$Grid\_Points(g) = \{< g, j, k > | \\ \forall j \leq Jmax(g) \wedge \forall k \leq Kmax(g) \\ (p_{j,k} = point(g, j, k))\}$$

$$Grid\_LineSegments(g) = \{< g, j, k > | \\ \forall j \leq Jmax(g) \wedge \forall k \leq Kmax(g) \\ (line\_segment(g, p_{i,j}, p_{i,j+1})) \\ (line\_segment(g, p_{i,j}, p_{i+1,j}))\}$$

Figure 8 The sensors for each grid (and control line as well) consist of a set of points and the lines which connect the points.

cannot be generated via the grid controller and SURGRD. Since all of the grids generated via the grid controller have constant kmax values (number of horizontal lines), to achieve similar results as PEGASUS, the grid controller drastically varies the etamx (grid line lengths) on portions of the farfield in order to achieve proper overlap.

The Seam Grid [10] work presents a solution for automatically decomposing and generating structured surface grids. The concept of a seam grid is introduced to define those hyperbolic grids close to seam or control lines. These seam grids are intended to cover the discontinuous regions of a CAD geometry. Given a set of seam grids, an algebraic scheme is utilized to cover the CAD geometry, stitching the algebraic grids together with the seam grids. SEAMCR [9] extends the seam grid system by developing an automated method for generating the seam lines by identfying qualitative features in a CAD geometry. Both [10, 9] rely on an algebraic scheme using the tool SBLOCK to connect the seam grids over smooth regions of the CAD geometry. The grid controller differs in that it attempts to fully cover the CAD geometry using grids grown from the seam lines and does not rely on algebraic grids to connect the seam grids. It marches the seam grids until undesirable grid qualities such as offbody or negative jacobians are encountered. The method the grid controller uses to determine the marching distances of the surface directly addresses the unsolved problem of determining the marching distances for the seam grids [9]. The grid controller should utilize the SEAMCR tool to generate control lines.

4.3 Other Systems

Adaptive structured grid generation methods such as Adaptive Mesh Refinement (AMR) [3] and Adaptive Spatial Partitioning and Refinement(ASPaR) [33] involve the generation and removal of structured grids at different levels of refinement to address "estimates of numerical error or regions of high gradients" which are unacceptable. The grid controller cannot add or remove grids, it can only modify the grids. In addition the grid controller does not modify grids in response to the results of simulations. It operates exclusively *before* simulations are run.

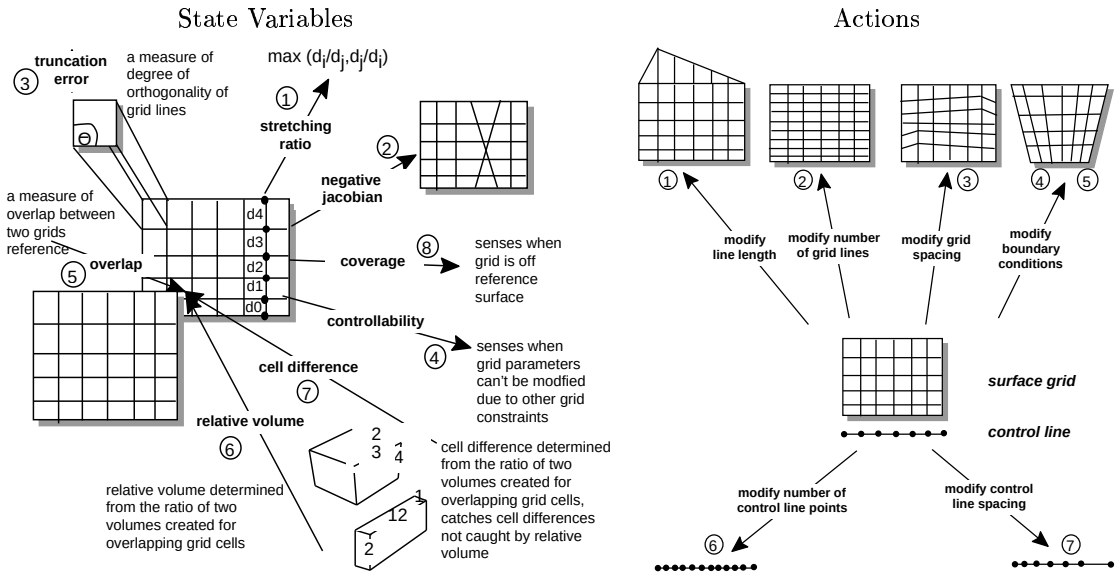


Figure 9 The figure on the left shows the eight state variable parameters or grid qualities used to characterize Overset structured grids. The figure of the right shows the seven actions or grid actions available to the grid controller. Section B details each of the grid qualities and grid actions.

The block structured multi-zonal methods of Eiseman [21] automatically generate block structured grids. Block structured grids however must abut each other such that all grid lines in one grid meet grid lines in another. This requirement significantly increases the number of points required for block structured methods. For a complex CAD geometry, there may be an order of magnitude larger number of points than the same set of grids from the Overset approach. This is due to the fact that when gridding complex geometries with great disparity in spacing requirements, Overset methods can overlap grids with varied grid size.

The Amphion [44] program synthesis system was applied to the domain of Overset grid generation.[5] The program synthesis system [47] generated input specifications for Chimera grid tools by reasoning over symbolic propositions defined for the geometric features of aerospace designs and Overset structured surface grids. The system was never made operational due in part to the challenges faced of integrating the program synthesis *symbolic* reasoning system with *numeric* grids and generation systems. Specifically, the symbolic geometric propositions did not have well defined mappings to the vast amount of numeric information represented by the grids and CAD aerospace model. In addition, the symbolic specifications generated by the program synthesis system were never grounded in numeric specifications required by the grid generation tools.

5 Expert System: Conceptual Description

The algorithm which implements the feedback control grid generation system is defined in Fig. 11. Both the control line and surface grid marching modes of the controller use iterative repair methods to refine the control lines and surface grids respectively. Their iterative repair strategies on each iteration differ significantly. In control line mode, on each iteration, the

[5] Team members: Dr. Richard Waldinger, Tom Pressburger, Dr. Steve Roach, Dr. Michael Lowry, Dr. William Van Dalsem, Dr. Goetz Klopfer, Dr. Yehia Rizk

Grid Control Laws		State Variables = Grid Qualities							
		stretching ratio	jacobian	truncation error	control-lability	overlap	relative volume	cell difference	coverage
Grid Actions	kmax	k_{11}	k_{12}	k_{13}	k_{14}	k_{15}	k_{16}	k_{17}	k_{18}
	etamx	k_{21}	k_{22}	k_{23}	k_{24}	k_{25}	k_{26}	k_{27}	k_{28}
	deta	k_{31}	k_{32}	k_{33}	k_{34}	k_{35}	k_{36}	k_{37}	k_{38}
	dfar	k_{41}	k_{42}	k_{43}	k_{44}	k_{45}	k_{46}	k_{47}	k_{48}
	ibcja	k_{51}	k_{52}	k_{53}	k_{54}	k_{55}	k_{56}	k_{57}	k_{58}
	ibcjb	k_{61}	k_{62}	k_{63}	k_{64}	k_{65}	k_{66}	k_{67}	k_{68}
	jmax	k_{71}	k_{72}	k_{73}	k_{74}	k_{75}	k_{76}	k_{77}	k_{78}

Figure 10 The K gain matrix represents the declarative form of the grid control laws. The rows and columns represent the grid state variables and grid actions as defined in Fig. 9. The elements of the table are heuristic rules which define the first-order mappings from each state variable to each action. Section 6 details the grid generation application. Section 7 details the theory of heuristic feedback control (HFC). Section B details the heuristic rules.

whole control line is searched for the region of greatest truncation error violation. This region of greatest truncation error violation is refined in order to achieve the truncation error objective set by the user. On the other hand, in surface grid marching mode, on each iteration, each of the j_line lengths (vertical lines) are refined in order to achieve the overlap setpoint objectives and other constraints defined by the user.

The grid controller accepts a grid schedule (Fig. 7, left) consisting of an ordered set of grid generation tasks. It iterates through each task in the grid schedule (Fig. 11, line 1) for the defined number of iterations of the task (Fig. 11, line 2). On each iteration, it determines the current setpoints or reference signals (Fig. 11. line 3). For each active grid (Fig. 11, line 4), the grid qualities of both the grid and those it interacts with (Fig. 11, line 5) are determined. The algorithm then branches based upon the control mode in effect for each grid schedule task (Fig. 11, line 6). The first control mode is for control line modification (Fig. 11, line 7). Under this mode, the control line for the current grid is selected (Fig. 11, line 8). Control line control laws are applied to determine a new control line specification for SRAP by minimizing the difference between the grid setpoints and grid qualities (Fig. 11, line 9). The new SRAP specifications are collected (Fig. 11, line 10). The second and third control modes are for marching and repairing Overset surface grids (Fig. 11, line 11). They are invoked in a similar fashion to that of the control line mode, except that now a new surface grid specification for SURGRD is generated by the grid control laws (Fig. 11, line 12) attempting to minimize the difference between the grid setpoints and grid qualities. The new SURGRD specifications are also collected (Fig. 11, line 13). At the end of the current iteration (Fig. 11, line 15), all of the SRAP specifications are executed by calls to *actuate_control_lines* (Fig. 11, line 16), followed by the execution of all of the SURGRD specifications through calls to *actuate_surface_grids* (Fig. 11, line 17). SRAP specifications are executed before SURGRD specifications due to the fact that each surface grid is defined with respect to a control line.

6 Grid Control Laws: Expert System Heuristics

The grid control laws are embedded in the function calls *control_line_modifications* and *surface_grid_modifications* (Fig. 11, lines 9 and 12). These functions execute the control laws defined in Eqn. 1. The evaluation of Eqn. 1 defines the inference procedure of the expert system.

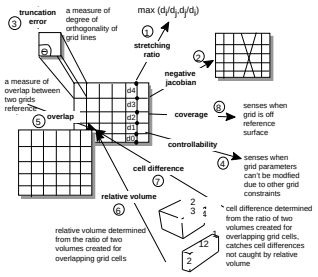
$$\vec{u}(t+1) = \mathbf{K}\vec{x}(t) \tag{1}$$

```
function Grid_Controller (schedule)
1   for task in schedule.tasks() do
2     for iteration in task.iterations() do
3       reference_signals = task.reference_signals()
4       for grid in task.grids() do
5         grid_qualities = sense_grid_and_dependencies(grid)
6         case control_mode
7           control_lines
8             control_line = grid.control_line()
9             control_line_spec = control_line_modifications(grid_qualities,reference_signals)
10            SRAP_specs += control_line_spec
11          surface_marching, surface_repair
12            surface_grid_spec = surface_grid_modifications(grid_qualities,reference_signals)
13            SURGRD_specs += surface_grid_spec
14        end case control_mode
15      end for grid
16      actuate_control_lines(SRAP_specs)
17      actuate_surface_grids(SURGRD_specs)
18    end for iteration
19  end for schedule
end function Grid_Controller
```

Figure 11 The grid control algorithm. The grid controller iteratively modifies control lines and surface grids until user specified reference signals are met. This is accomplished by the use of Overset grid tools SGQ, OLGQ, GRIDED, SRAP and SURGRD.

At each point in time, Eqn. 1 is solved for $\vec{u}(t+1)$ to provide the grid controller with the next set of synchronous grid actions to be taken. To understand how Eqn. 1 is evaluated requires an understanding of vectors $\vec{x}$, $\vec{u}$, matrix $\mathbf{K}$ as well as the machinery of matrix algebra. The elements of the vector $\vec{x}$ define the grid qualities in the grid control domain and are formalized as the state variable vector in Fig. 12. The elements of the vector $\vec{u}$ define the grid actions in the grid control domain and are formalized as the action vector in Fig. 13. The $\mathbf{K}$ matrix organizes the structure of the rule space. The row headers of the K matrix ennumerate the possible grid actions, and the column headers ennumerate the possible grid qualities (see Eqn. 2). The cartesian product of the row and columns yields the elements of the matrix. Each element of the matrix identifies a unique state/actuator combination implemented as a function or heuristic rule. Knowledge acquisition is performed by identifying the rows, columns and elements of the matrix. In the grid generation domain, the $\mathbf{K}$ matrix consists of 56 elements: 7 actions and 8 sensors.

$$\begin{bmatrix} u_{et} \\ u_{km} \\ u_{da} \\ u_{df} \\ u_{ia} \\ u_{ib} \\ u_{jm} \end{bmatrix} = \begin{bmatrix} k_{et,sr} & k_{et,nj} & k_{et,te} & k_{et,cn} & k_{et,ov} & k_{et,rv} & k_{et,cd} & k_{et,cv} \\ k_{km,sr} & k_{km,nj} & k_{km,te} & k_{km,cn} & k_{km,ov} & k_{km,rv} & k_{km,cd} & k_{km,cv} \\ k_{da,sr} & k_{da,nj} & k_{da,te} & k_{da,cn} & k_{da,ov} & k_{da,rv} & k_{da,cd} & k_{da,cv} \\ k_{df,sr} & k_{df,nj} & k_{df,te} & k_{df,cn} & k_{df,ov} & k_{df,rv} & k_{df,cd} & k_{df,cv} \\ k_{ia,sr} & k_{ia,nj} & k_{ia,te} & k_{ia,cn} & k_{ia,ov} & k_{ia,rv} & k_{ia,cd} & k_{ia,cv} \\ k_{ib,sr} & k_{ib,nj} & k_{ib,te} & k_{ib,cn} & k_{ib,ov} & k_{ib,rv} & k_{ib,cd} & k_{ib,cv} \\ k_{jm,sr} & k_{jm,nj} & k_{jm,te} & k_{jm,cn} & k_{jm,ov} & k_{jm,rv} & k_{jm,cd} & k_{jm,cv} \end{bmatrix} * \begin{bmatrix} x_{sr} \\ x_{nj} \\ x_{te} \\ x_{cn} \\ x_{ov} \\ x_{rv} \\ x_{cd} \\ x_{cv} \end{bmatrix} \quad (2)$$



$$\begin{bmatrix} strechingratio \\ jacobian \\ truncationerror \\ controllability \\ overlap \\ relativevolume \\ celldifference \\ coverage \end{bmatrix} = \begin{bmatrix} x_{sr} \\ x_{nj} \\ x_{te} \\ x_{cn} \\ x_{ov} \\ x_{rv} \\ x_{cd} \\ x_{cv} \end{bmatrix} = \vec{x} \text{ state variables}$$

Figure 12 The grid qualities vector $\vec{x}$ which define the possible grid qualities are formalized as the state vector $\vec{x}$. The elements of vector $\vec{x}$ correspond to the numbers in the diagram of the Figure. The dimension of vector $\vec{x}$ is [1 x 8].

Each element u_i of $\vec{u}$ is determined by the dot product of the k_ith row of $\mathbf{K}$ with the state vector $\vec{x}$. For example, action u_{et}, the first element of $\vec{u}$, is defined in Eqn. 3 as the dot product of the first row in the K matrix with the state vector.

$$u_{et} = \vec{k_{et}} * \vec{x} \tag{3}$$

Expanding Eqn. 3 yields Eqn. 4:

$$u_{et} \quad = \quad [k_{et,sr}, k_{et,nj}, \ldots, k_{et,cd}, k_{et,cv}][x_{sr}, x_{nj}, \ldots, x_{cd}, x_{cv}]^T \tag{4}$$

If the values of vector $\vec{x}$ are numeric, and the elements of the $\vec{k_{et}}$ are numeric constants, then the dot product operation will utilize the standard definitions of the mathematical functions plus $(+)$ and $(*)$ as seen in Eqn. 5.

$$u_{et} = k_{et,sr} * x_{sr} + k_{et,nj} * x_{nj} + k_{et,te} * x_{te} + k_{et,cn} * x_{cn} + k_{et,ov} * x_{ov} + k_{et,rv} * x_{rv} + k_{et,cd} * x_{cd} + k_{et,cv} * x_{cv} \tag{5}$$

Since all of the elements K matrix are functions or heuristic rules, the generalized dot product operation is defined as two layers of function calls (Eqns. 6,7).

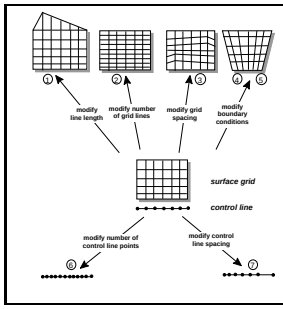
$$u_{et} = k_{et,sr}(x_{sr}) + k_{et,nj}(x_{nj}) + k_{et,te}(x_{te}) + k_{et,cn}(x_{cn}) + k_{et,ov}(x_{ov}) + k_{et,rv}(x_{rv}) + k_{et,cd}(x_{cd}) + k_{et,cv}(x_{cv}) \tag{6}$$

$$u_{et} = f(k_{et,sr}(x_{sr}), k_{et,nj}(x_{nj}), k_{et,te}(x_{te}), k_{et,cn}(x_{cn}), k_{et,ov}(x_{ov}), k_{et,rv}(x_{rv}), k_{et,cd}(x_{cd}), k_{et,cv}(x_{cv})) \tag{7}$$

To evaluate Eqn. 7, the mathematical function f is no longer interpreted as a *literal sum* of product terms but now is interpreted as an *abstract combination* of product terms. In addition, the mathematical functions $k_{et,*}$ are no longer interpreted as the *product* of numeric terms, but must now are interpreted as a generalized function application with x_* as the input arguments. The generalized dot product function is constructed by recognizing that the form of the dot product defined in Eqn. 7 is a function with sixteen arguments. The sixteen arguments are made up of eight arguments for the grid state variables and eight function arguments for the elements of the first row of the K matrix as can be seen in Eqn. 8.

$$u_{et} = f(k_{et,sr}, k_{et,nj}, k_{et,te}, k_{et,cn}, k_{et,ov}, k_{et,rv}, k_{et,cd}, k_{et,cv}, x_{sr}, x_{nj}, x_{te}, x_{cn}, x_{ov}, x_{rv}, x_{cd}, x_{cv}) \tag{8}$$

An infinite number of function forms can fit function f. The art of the knowledge engineer is to define those function forms appropriate to the domain. In the grid generation domain, two generalized dot product function forms have been explored. The first generalized dot product function form is implemented as a cascading conditional which ranks the grid qualities in terms of state and setpoint criticality (Fig. 14, left). The value of the action u_{et} is determined



$$\begin{bmatrix} line\_length : etamx \\ number\_h\_lines : kmax \\ initial\_spacing : deta \\ final\_spacing : dfar \\ left\_splay : ibcja \\ right\_splay : ibcjb \\ number\_v\_lines : jmax \end{bmatrix} = \begin{bmatrix} u_{et} \\ u_{km} \\ u_{da} \\ u_{df} \\ u_{ia} \\ u_{ib} \\ u_{jm} \end{bmatrix} = \vec{u} \text{ actions}$$

Figure 13 The grid actions $\vec{u}$ which define the possible actions are formalized as the action vector $\vec{u}$. The elements of vector $\vec{u}$ correspond to the numbers in diagram of grid actions. The dimension of vector $\vec{u}$ is [1 x 7].

by selecting the action associated with the first grid quality which satisfies its state variable predicate. The second generalized dot product function form is implemented as the minimum or intersection operation (Fig. 14, right). The value of the action u_{et} is determined by selecting the intersection (or mimimum) action as suggested by each of the product terms. The tradeoffs of generalizing the dot product are a loss of linear control law properties including controllability, observability and stability. The benefit is the ability to leverage the linear algebra machinery for non-linear heuristic closed-loop control domains.

6.1 Gain Matrix Elements (k_{ij}): Solving for u_{ij}

Each element of the K matrix encodes a local heuristic relating a single state variable to a single action. By identifying the heuristic for each element, the local heuristics for the grid controller expert system are systematically defined. Fig. 15 shows the pseudocode corresponding to the value of the $u_{et,cv} = k_{et,cv}(x_{cv})$ element. Note that the logic derives different actions depending upon whether the state variable coverage (x_{cv}) is *onbody*, *offbody*, or *just_offbody*. This results in a local control action u_{et} of *increasing_line_length*, *decreasing_line_length* or *maintaining_current_line_length* respectively. All other elements of the K matrix are defined in a similar fashion. The definitions for most elements can be found in Appendix B.

7 Heuristic Feedback Control (HFC) Theory

> We must be systematic, but we should keep our systems open.
> - Alfred North Whitehead, Modes of Thought

The theory of heuristic feedback control (HFC) combines the feedback properties of expert system controllers [27, 16] with the parallel execution paths of subsumption [6]. It combines production rules [18] of rule-based systems with linear algebra vector/matrices of modern control theory [5]. It follows the challenge outlined in Planning and Control [20] to develop methods of integrating traditional control and AI systems.

7.1 Define Variables for Objectives, Sensors, States and Actions

HFC leverages the methodology used in modern control theory [29] to design process controllers. The methodology defines a fixed vocabulary of variables and relations between variables. Four classes of vectors are required to develop process control applications. In a similar manner to

$f(\, k_{et,sr}, \ldots, k_{et,cv}, x_{sr}, \ldots, x_{cv}\,)$	$f(\, k_{et,sr}, \ldots, k_{et,cv}, x_{sr}, \ldots, x_{cv}\,)$
/* Version 1. Compute next *et.**/	/* Version 2. Compute next *et.**/
if $x_{nj} =$ *negative* then $k_{12}(x_{nj})$	minimum $\big($ $k_{11}(x_{sr})$,
elseif $x_{cv} =$ *off* then $k_{15}(x_{cv})$	$k_{12}(x_{nj})$,
elseif $x_{cn} \neq 0$ then $k_{14}(x_{cn})$	$k_{13}(x_{te})$,
elseif $x_{ov} \leq 2$ then $k_{18}(x_{ov})$	$k_{14}(x_{cn})$,
elseif $x_{rv} \leq .5$ then $k_{16}(x_{rv})$	$k_{15}(x_{cv})$,
elseif $x_{sr} \leq 1.3$ then $k_{11}(x_{sr})$	$k_{16}(x_{rv})$,
elseif *previous_et* exists then *previous_et*	$k_{17}(x_{cd})$,
elseif *observed_et* exists then *observed_et*	$k_{18}(x_{ov})\big)$
else *et_initial_guess*	end *functionf*
endif	
end *functionf*	

Figure 14 Two dot product functions for computing $u_{et} = f(k_{et,sr}, \ldots, k_{et,cv}, y_{sr}, \ldots, y_{cv})$ are defined. The first instance defines a function form which orders the state variables as a cascading conditional indicating the priority of the state variable, selecting the k_{1j} term whose state variable predicate is the first to match. The second instance defines a function form which computes the minimum of all contributions of k_{1j} terms.

process controller designers, closed-loop control expert system developers must exhaustively ennumerate the vectors corresponding to objectives ($\vec{r}$), sensors ($\vec{y}$), state variables ($\vec{x}$) and actions ($\vec{u}$) in the domain. For heuristic domains, variable values will be a mix of numeric and symbolic values. To define the variables and their values, the following questions must be answered for each domain:

- What are the objectives in the domain, and possible values for each objective?
  $\vec{r}(t) = \left[\ r_1(t) \cdots r_o(t)\ \right]$
- What are the sensors in the domain, and possible values for each sensor?
  $\vec{y}(t) = \left[\ y_1(t) \cdots y_m(t)\ \right]$
- What are the state variables in the domain, and possible values for each state variable?
  $\vec{x}(t) = \left[\ x_1(t) \cdots x_p(t)\ \right]$
- What are the actions in the domain, and possible values for each action?
  $\vec{u}(t) = \left[\ u_1(t) \cdots u_r(t)\ \right]$

7.2 Determination of the Transfer Function or Gain Matrix

The HFC control laws are determined through the definition of the mapping between the state variables ($\vec{x}$) and the actions ($\vec{u}$) according to a set of objectives ($\vec{r}$). This mapping, also called the transfer function is defined by answering the following questions:

- What is the relationship between the values of state variable x_i and each action $\vec{u}$, given objective r_i?

The linear algebra form of the transfer function or gain matrix is represented in Eqn. 9. The objectives or setpoints $\vec{r}$ are no longer explict in Eqn. 9 but are embedded in the logic of each k_{ij} element. The function definitions of each k_{ij} encodes the setpoints into the logic of the heuristic rules. Such an approach is taken in modern control theory as well as in a class of reactive systems.

$$\vec{u} \quad = \quad -\mathbf{K}\vec{x} \tag{9}$$

Given that the dimensions of $\vec{x}$ are [1 x m] and the dimensions of $\vec{u}$ are [r x 1], then the matrix $\mathbf{K}$ is of dimensions [r x m]. The $r * m$ elements of the $\mathbf{K}$ matrix are highlighted by the expansion of the $\mathbf{K}$ in Eqn. 10.

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_r \end{bmatrix} = \begin{bmatrix} k_{1,1} & k_{1,2} & \cdots & k_{1,m} \\ k_{2,1} & k_{2,2} & \cdots & k_{2,m} \\ \vdots & \vdots & \cdots & \vdots \\ k_{r,1} & k_{r,2} & \cdots & k_{r,m} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \tag{10}$$

Each k_{ij} element (Eqn. 11) must be elicited from the domain expert. To define each k_{ij} element the expert system designer must answer for each i and j, the following question:

- What affect will the jth state variable have upon the ith action?

$$\begin{bmatrix} \vdots \\ u_i \\ \vdots \end{bmatrix} = \begin{bmatrix} & & \vdots & \\ \cdots & \cdots & k_{i,j} & \cdots \\ & & \vdots & \end{bmatrix} * \begin{bmatrix} \vdots \\ x_j \\ \vdots \end{bmatrix} \tag{11}$$

Since the jth state variable and ith action each can take on a set of values, the answer to this question is a function of the form: $k_{ij}(x_j) = u_{ij}$. Depending upon the types of state variables and actuator variables, this function can be a mathematical construct, heuristic rules, table lookup or other mapping preserving form which follows naturally from the domain. By systematically defining each element in the matrix a first order heuristic feedback control system can be defined. An example from the grid generation domain can be seen in Fig. 15.

Once each k_{ij} element has been defined, the domain expert must define global heuristics for combining the local heuristics in order to solve the equation for each u_i of $\vec{u}$. To define the global heuristics of combining the local evidence the expert designer must answer for each u_i the following question:

- How will the local heuristics for the ith row of the $\mathbf{K}$ matrix, with respect to each x_j, be combined to define a net u_i value?

$$\begin{bmatrix} \vdots \\ u_i \\ \vdots \end{bmatrix} = \begin{bmatrix} & & \vdots & \\ k_{i,1} & k_{i,2} & \cdots & k_{i,p} \\ & & \vdots & \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \tag{12}$$

Each element of the actuator vector u_i can be derived via the dot product of the state variable vector $\vec{x}$ with ith row of the $\mathbf{K}$ matrix:

$$u_i = [k_{i,1}, k_{i,2} \ldots k_{i,p-1}, k_{i,p}][x_1, x_2 \ldots x_{p-1}, x_p]^T \tag{13}$$

Since each $k_{i,j}$ is a function call, this function u_i is defined as the sum of product terms:

$$u_i = k_{i,1}(x_1) + k_{i,2}(x_2) + \cdots + k_{i,p}(x_p) \tag{14}$$

Generalizing Eqn. 14 yields the abstract dot product function:

$$u_i = f(k_{i,1} \ldots k_{i,p}, x_1 \ldots x_p) \tag{15}$$

defrule $k_{et,cv}(x_{cv})$
/* Define the heuristic rule element $k_{et,cv}$ of the K matrix.*/
/* Coverage setpoint: total coverage*/
/* Effect of state variable coverage on action etamx */
/* When the grid goes offbody then return the last best onbody value and terminate.*/
if cv
onbody **then** et = et + Δ
offbody **then** et = et - Δ
onbody_and_last_iteration_offbody **then** et = et
endif
end $k_{et,cv}$

Figure 15 The heuristic rule definition of the k_{15} element for the K matrix. This element is a local grid heuristic which captures the relationship between the grid quality coverage (x_{cv}) and the grid action line length (u_{et}).

The more general definition of the dot product illustrates that the dot product function as defined in linear algebra is just one way of computing the value of u_i. Many other function definitions that utilize the same arguments as f, including non-linear function forms, can be defined. These function forms could take the form of a complex conditional to capture the priority, criticality and system state information sources, or it could take on the form of intersection of system state information sources where it accepts a minimum action which satisifies all components. Two such examples have been illustrated in the grid generation domain (Fig. 14).

The use of domain specific knowledge to implement the dot product function rather than relying upon the domain independent dot product definition of linear algebra, is an instance of Allan Newell's and Edward Feigenbaum's insight of the fundamental tradeoff between a problem solvers generality and its expertness. Newell states: "...there is an inverse relationship between the generality of a method and its power" [36] (page 12). Feigenbaum identifies a "law of nature operating that relates problem solving generality (breadth of applicability) inversely to power (solution success..) and power directly to specificity (task-specific information)" [22] (page 6). These principles are observed in the grid generation domain by providing powerful, narrowly applicable domain specific dot product heuristics which yield acceptable solutions in the grid generation domain.

7.2.1 K Matrix Extensions

As knowledge engineers start to acquire, represent and execute the expert system (via evaluation of Eqn. 9), they will find that some actions cannot be defined as the additive composition of state variable contributions. It is natural for the knowledge engineer to assume the simplest possible model, namely a linear model of control, and only add in higher order terms when nonlinearities are encountered. The process for the development of heuristic systems is analogous to that of a mathematical modeller adding in higher order Taylor series terms when appropriate. The K matrix can be expanded by adding additional rows to the K matrix to represent control laws which require conjunctive evidence. The state vector $\vec{x}$ can be expanded by adding additional elements to the state vector $\vec{x}$ to represent the conjunctive evidence.

8 Related Work: AI

HFC is a knowledge-based system defined by a set of propositions which have a truth value, and a theory which can reason over the propositions [31, 42]. The propositions are defined over the values of the $\vec{x}$, $\vec{u}$ and $\vec{r}$ vectors. The theory which reasons over the propositions is defined by the **K** matrix. At any point in time, working memory consists of propositions describing the domain variables and their values in the domain. No accumulation of historical data is permitted; only previous and current data are kept. The feedback control cycle of the expert system is defined by successive iterations over a fixed sequence of tasks. These tasks define a simple non-monotonic reasoning system, where historical information is constantly retracted while new information is asserted (see Fig. 16).

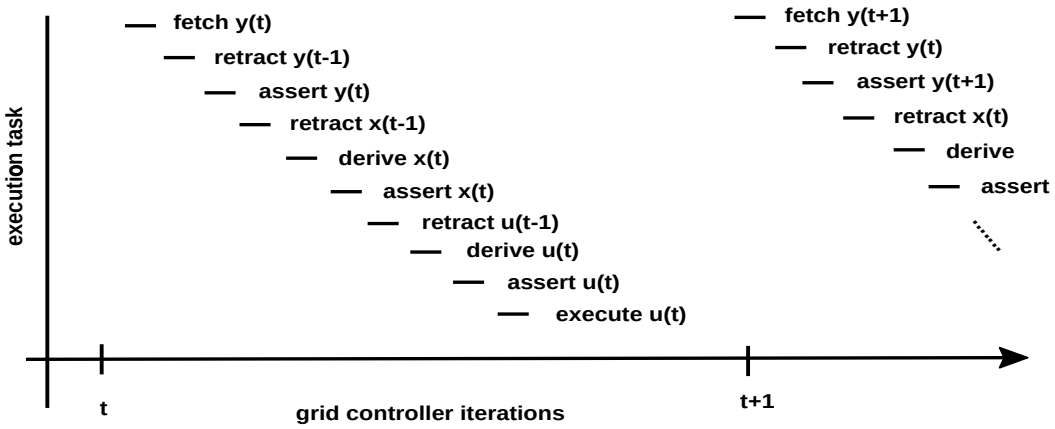


Figure 16 A fixed sequence of tasks is executed on each iteration of the grid controller.

HFC has its roots in the development of iterative feedback closed-loop systems. Closed-loop feedback control systems can be partitioned in conventional and AI control systems. Jörg Müller summarizes the commonalities and differences between conventional control and AI control:

> ...the basic view of the controller as consisting of a state estimation function and a regulation function...can be transformed [from the conventional control model] into the AI perspective on agents and their environment...the basic difference between the two methods are...that models [which compute] the effects of actions on the world [are] realized by differential equations in [conventional control] and by symbolic reasoning on explicitly represented aspects of the world in [AI based control] [34] (page 12).

Norbert Wiener, who defined cybernetics as "the science of control and communication in the animal and machine" [50] formalized the theory of feedback control. Sowa in Conceptual Structures, critiques cybernetics as not applicable to AI symbolic methods:

> Cybernetics is descended from analog control systems, but AI is based upon digital computer programming. Because of their heritage they use different formalisms: cybernetics uses continuous mathematics, especially calculus and differential equations; but AI uses discrete mathematics, especially symbolic logic and formal grammars...Although Weiner's original definition of cybernetics claimed the entire subject matter. [43] (page 340)

Both Müller and Sowa reject the methods of conventional control due to its reliance on continuous mathematics. This paper has demonstrated that a carte blanche rejection of the machinery of continous mathematics for heuristic systems is not warranted. HFC has shown that the linear algebra framework which often is used to model sets of first order differential equations can be generalized to allow for "symbolic reasoning on explicitly represented aspects of the world" [34] (page 12). This is accomplished by defining qualitative mathematical operations [48], which are characterized by symbolic mathematical reasoning over finite fields. In addition it is accomplished by abstracting from the language of mathematics to the language of set operations which underlie the mathematics. For example, the execution of the equation $\vec{u} = \mathbf{K}\vec{x}$ is defined as a sequence of matrix dot product operations. To determine an element of $\vec{u}$, the dot product operation is performed with a row of the $\mathbf{K}$ matrix and the column of the $\vec{x}$ vector. The application of each dot product operation can be reduced to a fixed number of set mapping relations between elements of the set of state variables $\vec{x}$, elements of the set of a row of the $\mathbf{K}$ matrix and elements of the set of actions $\vec{u}$ (see Fig. 17).

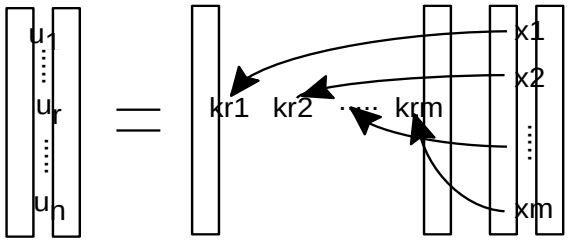


Figure 17 Systematic set mapping operations which underlie matrix dot product operations are the foundation for heuristic feedback control (HFC).

8.1 Conventional and AI Feedback Control Systems

The theory of heuristic feedback control is an attempt to "bring cog sci and AI back to the broader view of cognition and feedback mechanisms found in earlier psychology and cybernetics" [14] (page 132).

8.1.1 Modern Control Theory

HFC utilizes the variable type definitions, the matrix equations and the matrix operations from modern control theory [5, 38]:

$$x' = Ax + Bu \tag{16}$$

$$y = Cx \tag{17}$$

$$u = Kx \tag{18}$$

HFC uses matrices to represent the control knowledge of the expert system. The ease of organization of matrices makes them attractive for knowledge representation requirements. In addition, the use of linear algebra to develop expert systems in scientific and engineering domains is a natural fit due to the fact that matrices and vectors are already fundamental tools of scientists and engineers.

The K matrix is analogous to a multi-input/multi-output transfer function of frequency based methods [32] and the gain K matrix of modern control theory [5]. The $\mathbf{K}$ matrix is extended to allow for rule elements and not just constant elements. These equations are extended

SystemName	*Feature*1	*Feature*2	*Feature*3	*Feature*4
ModernControl Theory[5]	$\vec{y}$	$\vec{x} = \mathbf{C}^{-1}\vec{y}$	$\vec{u} = \mathbf{K}\vec{x}$	$\vec{u}$
ExpertSystem Controller[27]	*monitoring*	*interpretation*	*prediction*	*repair*
Mycin[18, 11]	*monitor*	*diagnose*	*identify*	*modify*
Subsumption[6]	*fetch_sensors*		*activate_layers*	*inhibit_suppress*
GridController	*fetch_points* *fetch_lines*	*determine* *grid_qualities*	*compute* *g_cl_specs*	*execute_SRAP* *execute_SURGRD*

Figure 18 A comparison of closed-loop feedback control systems including traditional control as well as expert system controllers. All of the systems share a common feedback control cycle. The systems differ in the degree of processing of the raw sensory information into internal variables, the amount of search utilized to determine the next actions as well as the degree of independence of parallel control loops.

through the use of symbolic quantities as variable values, and the use of the matrix dotproduct operations generalized from a literal sum of products to a more abstract combination of terms written in domain specific heuristics. This paper assumes that $\vec{x}$ is defined via the inverse of $y = Cx$, namely $x = C^{-1}y$. This inversion is implemented via domain specific deterministic methods. [6] Derivation is outside of the scope of this paper (see Appendix A).

Instead of the **K** matrix being defined through an optimization process, **K** is defined by interviewing domain experts. Knowledge acquisition for the **K** matrix is accomplished through interviewing the domain experts in order to systematically identify the **K** matrix elements. The use of matrices has many benefits. "The spreadsheet's visual metaphor [of the matrix] amplifies the expert's ability to recognize and offer distinctions between the elements on the basis of the constructs" [23] (page 82). Each element of the array is a modular piece of knowledge defining an empirical association between a state variable and an actuator. *Missing matrix elements imply missing expert knowledge.* The ability to identify missing matrix elements allows us to remove the difficulty commonly associated with heuristic systems, namely: a "system based upon empirical associations is more difficult to construct because the character of the knowledge makes it necessary to extract the rules on a a case-by-case basis." [17] (page 391). Using matrices, it is easy to support knowledge acquisition from multiple domain experts. In the grid generation domain, three domain experts were used concurrently; each identified the rules of separate columns of the **K** matrix.

Similarities exist between the matrices used by HFC and those employed by a key knowledge acquisition tool in personal construct theory [4]. Matrices identified as repertory grids [30] are used to scale a set of elements with respect to a given personal construct. These types of matrices are defined as a set of similar-contrast pole assignments over triadic subsets of the elements. Such methods should be included in HFC when defining the finite valueclass and orderings of domain variables.

8.1.2 Early Expert Systems

Ten "generic categories of knowledge engineering applications" are defined in An Overview of Expert Systems [27]. One of the ten categories of knowledge engineering applications is defined as *control* knowledge engineering applications (CKEA). Such knowledge engineering

[6] SGQ and OLGQ: Jeff Onufer, Goetz Klopfer and Peter Robinson

applications are feedback control systems which "adaptively govern the overall behavior of a system" [27] (page 15). This is accomplished by repeatedly executing a feedback loop defined as a sequence of four sequential tasks: interpretation, prediction, repair and monitoring. Common operation between the grid controller and the four CKEA tasks can be seen in Fig. 18.

8.1.3 Heuristic Classification and Model-Based Reasoning

In Heuristic Classification [11], Mycin is defined as an instance of the class of *maintenance cycle* expert systems. This type of expert system iteratively executes four sequential tasks: monitor, diagnose, identify and modify. Common operation between the grid controller and the four maintenance cycle tasks can be seen in Fig. 18.

The heuristic rules [11] which comprise the elements of the K matrix of the form: **if** x_j **then** u_i are instances of heuristic classification [11]. The rules draw non-hierarchical and non-definitional inferences between elements of $\vec{x}$ and elements of $\vec{u}$. It is non-hierarchical because elements of $\vec{u}$ are not elements of the abstraction lattice of $\vec{x}$. It is non-definitional because the relations in $\mathbf{K}$ between $\vec{x}$ and $\vec{u}$ do not define the attributes of $\vec{x}$ or $\vec{u}$.

The K matrix can be modelled as a model-based reasoning (MBR) component [25] in which the column headers define the state variable ($\vec{x}$) inputs to the component and the row headers define the action ($\vec{u}$) outputs from the component. In MBR, the mapping between inputs and outputs is often defined using first principles, qualitative constraints, e.g ([x] + [y] = [z])[19]. In HFC the mapping between inputs and outputs of the matrix component is defined through table lookup. Each of the table elements is a heuristic classification rule. HFC leverages both the heuristic classification approach of capturing empirical associations and the model-based reasoning component paradigm. HFC views the debate of when/where to utilize heuristic classification [11] vs. first principles qualitative constraints [17] as analogous to the debate in mathematical modelling of when/where to utilize models defined as curve fits to empirical data vs. utilizing models defined from first principles.

The dot product rules can be viewed as operational rule models [15]. Rule models are "abstract descriptions of subsets of rules, built from empirical generalizations of these rules, and are used to characterize a typical member of a subset." [15] (page 231). The integration process required by the dot product rules abstracts not the form of each rule but the form of the results of each rule.

8.1.4 Subsumption

The grid controller also exhibits many properties of subsumption [7]. No search through a space of internal representations is allowed. Internal representations are allowed as long as the internal representation is a deterministic conduit from sensing to acting. Multiple goals are implemented as competing parallel paths of execution. Separate layers define each goal. Many layers compete to actuate a limited set of actuators due to the fact that actuators overlap with other goal/layers (Fig. 19). Each layer receives the required sensor information synchronously. Actuators commands are executed synchronously, often with idempotent action. i.e. action which maintains the current command. The definition of each layer is not implemented as augmented finite-state machine (AFSM) with timers, but instead is implemented as heuristic sense act rules with a global clock.

Each symbolic dot product operation can be modelled as a set of subsumption layers with an inhibition/suppression module [6] (See Fig 19).

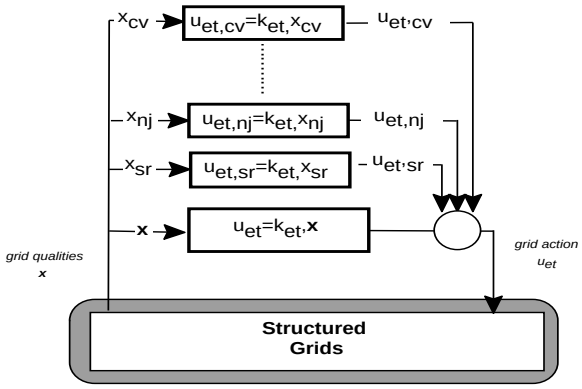


Figure 19 The grid controller can be modelled as a subsumption architecture. The diagram implements the subsumption model of $u_{et} = \mathbf{K}_{et}\vec{x}$.

8.2 Knowledge Acquisition

Many knowledge acquisition methods [12, 13, 26, 27, 41, 49, 51] fall short of providing the knowledge engineer the *types* of knowledge to look for. Though most of these knowledge acquisition systems model the world as a set of variables and relations between the variables (propositional and higher order), they are too general as to *what* knowledge to acquire and represent: "the typology of knowledge sources is too general. The precise meaning of the knowledge sources is ambiguous" [49] (page 113). In addition, these systems offer little help in ensuring that systematic coverage of the domain knowledge is achieved. Without strategies to identify the essential knowledge, it is impossible to define systematic knowledge elicitation and acquisition methodologies.

The KADS (Knowledge Analysis and Design Support) philosophy states that: "a system that has knowledge about the kinds of knowledge that it needs to acquire can exercise much more focused control on the acquisition process"[49] (page 114). Gaines [24] identified "practical reasoning knowledge" as an instance of KADS modelling. "Practical reasoning knowledge" specifically looks for feedback control processes in the behavior of domain experts. The grid generation domain was found by making explicit the feedback control processes of grid users. Gaines, however, does not link these knowledge acquisition models with the feedback control methodology used by process control designers to develop feedback controllers. HFC links practical reasoning feedback control knowledge with the knowledge acquisition, representation and execution methods of process controllers [38] (Chapter 22). This methodology explicitly models both the controller and its environment. It outlines eight stages for the development of feedback control systems. Explicit models of the controlling and controlled process are developed. Though the methodology was developed for process control applications, nothing in the methodology precludes the parameters and their relations from being symbolic, qualitative terms rather than quantitative, numeric terms. It is a general methodology which HFC leverages to model human, domain expert feedback behavior. Frequency-based [32] and time-based [5] methods can be used to implement the methodology. We have chosen the time-based methods with its use of matrices to naturally represent multiple input/output control functions.

8.3 Concept Maps and Concept Graphs

Concept maps [1] are useful to model hierarchical concepts, ranking the concepts from the "most general, most inclusive concept to the most specific, least general concept" [37] (page

1). They can be used to model those aspects of the **K** matrix rules which are hierarchical in nature. However, concept maps provide no guidance as to what concepts to look for and provide no methods to extract a formal structure for the concept maps to reason over.

Concept graphs are defined as "a directed bipartite graph composed of concepts and relations" [43] (page 70). Concept graphs can be used to model rule elements of the K matrix, however, the clarity of thought gained from viewing the matrix as a whole is lost.

9 Conclusions and Future Work

This paper has introduced the grid generation expert system application, as well as the theory of heuristic feedback control (HFC) for developing such expert systems. The grid generation system has achieved some success in the automation of Overset structured grid generation. Much tuning of the system will be required before the system is baselined for an aerospace design project. The introduction of adaptivity and learning into the grid controller will allow for runtime modification of ineffective grid heuristics. The grid heuristics need to be tested on additional and increasingly more complicated aerospace CAD design geometries. Additional grid controller modes which interleave control line mode with surface marching mode will address overlap issues related to disparate grid cell sizes. Additional global constraints on grid actions such as Vinokur's function [46] need to be integrated into the K matrix evaluation process. The utilization of additional knowledge, for example the aircraft grid script systems [40, 35], should be incorporated into the grid controller.

The experience of using methods from modern control theory [5, 38] has helped advance the use of process control methods in the developement of heuristic feedback control expert systems. It provides benefits over the full knowledge lifecycle of an expert system for knowledge acquisition, knowledge representation and knowledge execution. Modern control theory provides benefits during the *knowledge acquisition* phase by systematically identifying the classes of knowledge which need to be elicited in the domain. The process of systematically exploring all of the relationships between the grid qualities and the grid actions lead naturally to consider possibilities outside the scope of normal best practices for grid users. It provides benefits for *knowledge representation* of the domain by relying upon matrices and vectors to organize classes of domain heuristic rules. Systematic coverage of knowledge is reduced to identifying the elements of matrices. Missing matrix elements imply missing heuristics. *Knowledge performance* benefits allow for constant time execution of the expert system. This is accomplished through symbolic dot product analogs based upon on the deterministic numeric dot product operation implemented as a fixed and constant number of rule matchings.

10 Acknowledgements

I would like to thank Dr. William Van Dalsem and Dr. Eugene Tu for having the foresight and perseverance to fund this collaboration between artificial intelligence technologists and computational aerodynamicists. This work was funded under NASA Aero IT Base (519). I would also like to thank Dr. Michael Lowry and Dr. Goetz Klopfer for their vision and support. In addition I would like to thank Jeff Onufer, Dr. Donovan Mathias and Dr. Yehia Rizk for their patience in explaining structured grid generation to me. In memory of Dr. Herbert Kay.

References

[1] R. Abrams, D. Kothe, and R. Iuli. Meaningful Learning: A Collaborative Literature Review of Concept Mapping. *www2.ucsc.edu/mlrg/clr-conceptmapping.html.*

[2] J. Benek, P. Buning, and J. Steger. A 3-D Chimera Grid Embedding Technique. *AIAA-85-1523*, 1985.

[3] M. J. Berger and J. Oliger. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *Journal of Computational Physics*, 53:484–512, 1984.

[4] J. Boose and J. Bradshaw. Expertise Transfer and Complex Problems: Using AQUINAS as a Knowledge Acquisition Workbench for Knowledge-Based Systems. *International Journal of Man-Machine Studies(26), 3-28.*, 1987.

[5] W. L. Brogan. *Modern Control Theory.* Prentice-Hall, 1982.

[6] R. Brooks. *Cambrian Intelligence.* The MIT Press, Cambridge, MA, 1999.

[7] R. A. Brooks. A Robust Layered Control System for a Mobile Robot. Technical Report 864, MIT, 1985.

[8] W. M. Chan. Overgrid Version 1.6 . *http://www.nas.nasa.gov/Main/Frontpage/Features/chimerasidebar.html*, 2000.

[9] W. M. Chan and R. J. Gomez. Advances In Automatic Overset Grid Generation Around Surface Discontinuities. In *AIAA-99-3303*, 1999.

[10] W. M. Chan and R. L. Meakin. Advances Towards Automatic Surface Domain Decomposition and Grid Generation For Overset Grids. In *AIAA-99-3303*, 1999.

[11] W. Clancey. Heuristic Classification. *Artificial Intelligence*, 27:289–350, 1985.

[12] W. Clancey. The Knowledge Engineer as Student: Metacognitive Bases for Asking Good Questions. In H. Mandl and A. Lesgold, editors, *Learning Issues for Intelligent Tutoring Systems*, pages 80–113. Springer, New York, 1988.

[13] W. Clancey. The Knowledge Level Reinterpreted: Modeling of Socio-Technical Systems. *International Journal of Intelligent Systems*, 8(1):33–49, 1993.

[14] W. Clancey. *Situated Cognition: On Human Knowledge and Computer Representations.* Cambridge University Press, 1997.

[15] R. Davis. Interactive Transfer of Expertise: Acquisition of New Inference Rules. *Artificial Intelligence*, 12:121–157, 1979.

[16] R. Davis. Reasoning from first principles in electronic troubleshooting. *International Journal of Man-Machine Studies*, 19:403–423, 1983.

[17] R. Davis. Diagnostic Reasoning Based on Structure and Behavior. *Artificial Intelligence*, 24:347–410, 1984.

[18] R. Davis, B. Buchanan, and E. Shortliffe. Production Rules as a Representation for Knowledge-Based Consultation Programs. In W. J. Clancey and E. H. Shortliffe, editors, *Readings in Medical Artificial Intelligence: The First Decade*, pages 98–130. Addison-Wesley, Reading, MA, 1984.

[19] J. de Kleer and J. S. Brown. A Qualitative Physics Based on Confluences. *Artificial Intelligence*, 24:7–83, 1984.

[20] Dean and Wellman. *Planning and Control.* Morgan Kaufmann, first edition, 1991.

[21] P. Eiseman. Multiblock Grid Generation with Automatic Zoning. *NASA Grid Generation Conference*, pages 143–162, 1995.

[22] E. Feigenbaum, B. Buchanan, and J. Lederberg. On Generality and Problem Solving: A Case Study Involving the Dendral Program. Technical Report CS176, Stanford Computer Science, 1971.

[23] K. Ford, F. Petry, J. Adams-Webber, and P. Chang. An Approach to Knowledge Acquistion Based on the Structure of Personal Construct Systems. *IEEE Transactions on Knowledge and Data Engineering*, 3(1), 1991.

[24] B. Gaines. Modeling Practical Reasoning. *International Journal of Intelligent Systems*, 8(1):51–70, 1993.

[25] W. Hamscher, L. Console, and J. de Kleer. *Readings in Model-Based Diagnosis.* Morgan Kaufmann, San Mateo, CA, 1992.

[26] F. Hayes-Roth. Towards Benchmarks for Knowledge Systems and Their Implications for Data Engineering. *TKDE*, 1(1):101–110, 1989.

[27] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat. An Overview of Expert Systems. In F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, editors, *Building Expert Systems*, pages 3–29. Addison-Wesley, London, 1983.

[28] W. D. Henshaw. Automatic Grid Generation. *Acta Numerica*, pages 121–148, 1996.

[29] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering, Transactions of the ASME*, pages 35–45, 1960.

[30] G. Kelly. *The Psychology of Personal Constructs.* Norton, 1955.

[31] H. J. Levesque and R. J. Brachman. A Fundamental Tradeoff in Knowledge Representation and Reasoning (Revised Version). In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, pages 41–70. Kaufmann, Los Altos, CA, 1985.

[32] A. G. MacFarlane. *Frequency-Response Methods in Control Systems.* IEEE Press, New York, NY, 1979.

[33] R. L. Meakin. On Adaptive Refinement and Overset Structured Grids. In *AIAA-97-1858*, 1997.

[34] J. Muller. *The Design of Intelligent Agents.* Springer Verlag, 1996.

[35] S. Nash. Automating the CFD Process Using Overset Grids for a Wing/Fuselage/Nacelle Configuration. *NASA Grid Generation Conference*, 1994.

[36] A. Newell. Heuristic Programming: ILL-Structured Problems. In P. S. Rosenbloom, J. E. Laird, and A. Newell, editors, *The Soar Papers: Research on Integrated Intelligence (Volume 1)*, pages 3–54. MIT Press, London, 1993.

[37] J. D. Novak. The Theory Underlying Concept Maps and How To Construct Them. *http://cmap.coginst.uwf.edu/info/.*

[38] R. H. Perry, D. W. Green, and J. O. Maloney, editors. *Perry's Chemical Engineers' Handbook*. McGraw-Hill Book Company, sixth edition, 1984.

[39] P. Robinson. Feedback To Basics. In *Proceedings of the 1997 AAAI Fall Symposium on Model-Centered Autonomous Systems*, 1997.

[40] S. E. Rogers, K. Roth, S. Nash, M. D. Baker, J. P. Slotnick, M. Whitlock, and H. V. Cao. Advances in Overset CFD Processes Applied to Subsonic High-Lift Aircraft. *AIAA-2000-4216*, 2000.

[41] M. Shaw and B. Woodward. Modeling Expert Knowledge. *Knowledge Acquisition*, 2(3), 1900.

[42] B. C. Smith. Prologue to "Reflection and Semantics in a Procedural Language". In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, pages 31–39. Kaufmann, Los Altos, CA, 1985.

[43] J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[44] M. Stickel, R. Waldinger, M. Lowry, T. Pressburger, and I. Underwood. Deductive Composition of Astronomical Software from Subroutine Libraries. In *Proceedings of the Conference on Automated Deduction*, 1994.

[45] N. E. Suhs, W. E. Dietz, S. E. Rogers, S. M. Nash, and J. T. Onufer. PEGASUS 5.1. *http://www.nas.nasa.gov/~rogers/pegasus/uguide.html*, 2000.

[46] M. Vinokur. On One-Dimensional Stretching Functions for Finite-Difference Calculations. *Journal of Computational Physics*, pages 215–234, 1983.

[47] R. Waldinger. Amphion Applied to Computational Fluid Dynamics - unpublished report, 1995.

[48] D. S. Weld and J. de Kleer, editors. *Readings in Qualitative Reasoning About Physical Systems*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[49] B. J. Wielinga, A. T. Schreiber, and J. A. Breuker. KADS: A Modelling Approach to Knowledge Engineering. *Knowledge Acquisition*, 4:5–53, 1992.

[50] N. Wiener. *Cybernetics or Control and Communication In the Animal and the Machine*. MIT Press, Cambridge, 1948.

[51] L. Wood and J. Ford. Structuring Interviews with Experts During Knowledge Elicitation. *International Journal of Intelligent Systems*, 8(1):71–90, 1993.

A Grid Qualities: State Variables($\vec{x}$)

Grid quality routines provide quantitative assessments of what were previously qualitative visual assessments. Modern control theory links the observations ($\vec{y}$) to the state vaiables ($\vec{x}$) through the equation $\vec{y} = \mathbf{C}\vec{\mathbf{x}}$. Inversion of this equation yields $\vec{x} = \mathbf{C^{-1}}\vec{\mathbf{y}}$. The grid controller relies on routines OLGQ, and SGQ to implement this inverse.[7] These programs analyze the grid points produced by the grid generators as well as analyze the performance of the grid generators. The operation of the programs SGQ and OLGQ are not the focus of this paper.

Eight different grid qualities are utilized by the grid controller. The eight different grid qualities are divided into two categories: unary grid measures and binary grid measures. The single grid measures are stretching ratio, truncation error, jacobians, coverage and controllability. The binary grid measures are overlap, relative volume and cell difference.

Stretching ratio is a measure of change of distance between any three consecutive points. Two such measures are used by the controller, a local measure and a global measure. The local measure is defined between any three consecutive points as the maximum ratio between the two distances. A global stretching ratio keeps track of the minimum and maximum values as the three point window slides over a grid line with n points.

Jacobians are a measure of how close grid lines are to crossing. When a jacobian value is negative then grid lines have crossed. Grids with such conditions cannot be used for simulations.

Truncation error is a measure of the degree of orthogonality of a grid volume. Truncation error can be separated into solution-induced truncation error and mesh-induced truncation error. The truncation error grid quality is defined as the coefficient of the mesh-induced truncation error. A global measure for the truncation error for a whole grid is determined as the root-mean squared (RMS) point truncation error.

Controllability is defined as the difference between the grid specification as provided as input to a grid generator, and the measure of the grids after they have been generated. It was discovered that often the failure of the grid controller to achieve its desired results was due to constraints imposed by the grid generator itself. The information from controllability will be required to develop an adaptive grid controller.

Grid coverage is a measure of how much a grid covers a reference surface, covers another grid and how much a grid is offbody. Offbody coverage occurs when a grid has been advanced too far and it leaves the reference surface.

Overlap is a binary grid quality which measures the amount of overlapping grid cells between a recipient and donor grids. It is a grid quality measure which when met will help ensure that proper interpolation stencils can be defined. Overlap between recipient grid point and a donor grid can be measured with respect to four direction donor grid boundaries: kmin, kmax, jmin and jmax. An overlap state is defined for each of these four directions. An overlap state can take on one of four values, namely: 1 implies no overlap, 2: not enough, 3: enough, 4: too much.

Relative volume is a binary grid quality which measures the ratio of the volumes between recipient and donor grid cells created from overlapping surface grid cells.

Cell difference is a binary grid quality which measures the ratio of the shapes between recipient and donor grid cells created from overlapping surface grid cells. For example, the volume of recipient and donor grid cells could be equal but the recipient and donor cells could be of drastically different shape (i.e. cube vs. other parallelepiped).

[7] Jeff Onufer, Goetz Klopfer and Peter Robinson

B K Matrix Element Definitions

Each element of the K matrix is a function/heuristic rule which accepts a single sensor as input and produces an action command as output. All of the grid knowledge embodied in these heuristics was defined through interviews and queries with four domain experts from NASA Ames: Dr. Goetz Klopfer, Dr. Donovan Mathias, Jeff Onufer and Dr. Yehia Rizk.

defrule k_{11} (sratio)
   /* Effect of sratio on etamx with sratio objective of 1.3*/
   if sratio
      $\leq$ 1.3 **then** etamx = etamx
      $>$ 1.3 **then** etamx = etamx - Δ
   endif
end k_{11}

defrule k_{21} (sratio)
   /* Effect of sratio on deta with sratio objective of 1.3*/
   if sratio
      $\leq$ 1.3 **then** deta = deta
      $>$ 1.3 **then** deta = deta - Δ
   endif
end k_{21}

defrule k_{31} (sratio)
   /* Effect of sratio on dfar with sratio objective of 1.3*/
   if sratio
      $\leq$ 1.3 **then** dfar = dfar
      $>$ 1.3 **then** dfar = dfar - Δ
   endif
end k_{31}

defrule k_{41} (sratio)
   /* Effect of sratio on kmax with sratio objective of 1.3*/
   if Count ((from j = 1 to jmax) sratio $<$ 1.3)
     $<$ 50% **then** kmax = kmax + Δ
     $>$ 50% **then** kmax = kmax - Δ
   endif
end k_{41}

defrule k_{12} (jacobian)
   /* Effect of jacobian on etamx with jacobian objective of greater than 0*/
   if jacobian
      $\leq$ 0 **then** etamx = etamx - Δ
      $>$ 0 **then** etamx = etamx
   endif
end k_{12}

defrule k_{22} (jacobian)
   /* Effect of jacobian on deta with jacobian objective of greater than 0*/
   /* only relevant if negative jacobian occurs near control line or kmin line*/
   if jacobian
      $\leq$ 0 and (k $\approx$ kmin) **then**$\forall j$ deta = deta -Δ
      $>$ 0 **then** deta = deta - Δ
   endif
end k_{22}

defrule k_{32} (jacobian)
   /* Effect of jacobian on dfar with jacobian objective of greater than 0*/
   /* only relevant if negative jacobian occurs near kmax line */
   if jacobian
      $\leq$ 0 and (k $\approx$ kmax) **then**$\forall j$ dfar = dfar -Δ
      $>$ 0 **then** dfar = dfar - Δ
   endif
end k_{32}

defrule k_{42} (jacobian)
   /* Effect of jacobian on kmax with jacobian objective of 0*/
   /* The only reason to change this is if there is running the subiteration procedure*/
   /* which doubles the number of points to grow the grids*/
   /* then halves them after the grids are grown*/
   if jacobian
      $\leq$ 0 and (smoothing has failed)
      then kmax = kmax + Δ (double number of points)

rerun grid generator
kmax = kmax - Δ (halve number of points)
$>$ 0 **then** kmax = kmax
endif
end k_{42}

defrule k_{52} (jacobian)
/* Effect of jacobian on ibcja (right splay) with jacobian objective of 0*/
/* If NJ near center of control line, the increase splay, else reduce splay (if constraints for offbody permit it)*/
/* The change of splay is determined by how the negative jacobian is forming near the ends of the control lines. */
/* If grid lines are spiralling outward, then splay needs to be reduced. */
/* If grid lines are crossing into the interior of the grid, then need to increase the splay. */
/* I'm not sure how the outward spirals produce negative jacobians, */
/* but these are just as undesirable as standard negative jacobians. */
if jacobian
$\leq$ 0 and (j $\approx \frac{jmax}{2}$) **then** ibcja = ibjca + Δ
$\leq$ 0 and (or (j $\approx$ jmin) (j $\approx$ jmax)) **then** ibcja = ibjca - Δ
$\leq$ 0 and (spirals outward) **then** ibcja = ibjca - Δ
$\leq$ 0 and (crosses into interior of grid) **then** ibcja = ibjca + Δ
$>$ 0 **then** ibcja = ibcja
endif
end k_{52}

defrule k_{62} (jacobian)
/* Effect of jacobian on ibcjb (right splay) with jacobian objective of 0*/
/* See comments for k_{52} for comments. */
if jacobian
$\leq$ 0 and (j $\approx \frac{jmax}{2}$) **then** ibcjb = ibcjb + Δ
$\leq$ 0 and (or (j $\approx$ jmin) (j $\approx$ jmax)) **then** ibcjb = ibcjb - Δ
$\leq$ 0 and (spirals outward) **then** ibcjb = ibcjb - Δ
$\leq$ 0 and (crosses into interior of grid) **then** ibcjb = ibcjb + Δ
$>$ 0 **then** ibcjb = ibcjb
endif
end k_{62}

defrule k_{13} (truncation_error)
/* Effect of truncation error on etamx with truncation error objective of .005*/
/* Truncation error is proportional to cell area which increases as etamx increases*/
if truncation_error
$\leq$ .005 **then** etamx = etamx
$>$ .005 **then** etamx = etamx - Δ
endif
end k_{13}

defrule k_{23} (truncation_error)
/* Effect of truncation error on deta with truncation error objective of .005*/
/* Truncation error is proportional to cell area which decreases as deta increases*/
if truncation_error
$\leq$ .005 **then** deta = deta
$>$ .005 **then** deta = deta
endif
end k_{23}

defrule k_{33} (truncation_error)
/* Effect of truncation error on dfar with truncation error objective of .005*/
/* Truncation error is proportional to cell area which decreases as dfar increases*/
if truncation_error
$\leq$ .005 **then** dfar = dfar
$>$ .005 **then** dfar = dfar
endif
end k_{33}

defrule k_{43} (truncation_error)
/* Effect of truncation error on kmax with truncation error objective of .005*/
/* Truncation error is proportional to cell area which decreases as kmax increases*/
if truncation_error
$\leq$ .005 **then** kmax = kmax
$>$ .005 **then** kmax = kmax + Δ
endif
end k_{43}

defrule k_{14} (controllability)
/* Effect of etamx controllability on etamx with controllability objective of 0*/

```
    /* Relax the farfield around regions where controllability is detected
    /* This is performed in grid contorller repair mode
    if   controllability
         > 0 then etamx = etamx
         = 0 and (close to regions) > 0 then etamx = etamx + Δ
    endif
end k14

defrule k24 (controllability)
    /* Effect of etamx controllability on kmax with controllability objective of 0*/
    /* No effect*/
    kmax = kmax
end k24

defrule k34 (controllability)
    /* Effect of etamx controllability on deta with controllability objective of 0*/
    /* No effect*/
    deta = deta
end k34

defrule k44 (controllability)
    /* Effect of etamx controllability on dfar with controllability objective of 0*/
    /* No effect*/
    dfar = dfar
end k44

defrule k54 (controllability)
    /* Effect of etamx controllability on ibcja with controllability objective of 0*/
    /* No effect*/
    if   controllability
         > 0 and (j ≈ jmin) then ibcja = ibjca + Δ
         < 0 and (j ≈ jmin) then ibcja = ibjca - Δ
         = 0 then ibcja = ibjca
    endif
end k54

defrule k64 (controllability)
    /* Effect of etamx controllability on ibcjb with controllability objective of 0*/
    /* No effect*/
    if   controllability
         > 0 and (j ≈ jmax) then ibcjb = ibjcb + Δ
         < 0 and (j ≈ jmax) then ibcjb = ibjcb - Δ
         = 0 then ibcjb = ibjcb
    endif
end k64

defrule k15 (coverage)
    /* Effect of coverage on etamx with coverage objective of total coverage*/
    /* The heuristic used is to always remember the the far field distances */
    /* of the last point on the body. When the grid goes offbody then return */
    /* it to the last best onbody value.*/
    if   coverage
         onbody then etamx = etamx + Δ
         offbody then etamx = etamx - Δ
         onbodyandlastiterationoffbody then etamx = etamx
    endif
end k15

defrule k25 (coverage)
    /* Effect of coverage on deta with coverage objective of total coverage*/
    deta = deta
end k25

defrule k35 (coverage)
    /* Effect of coverage on dfar with coverage objective of total coverage*/
    dfar = dfar
end k35

defrule k45 (coverage)
    /* Effect of coverage on kmax with coverage objective of total coverage*/
    kmax = kmax
end k45
```

defrule k_{55} (coverage)
/* Effect of coverage on ibcja (right splay) with coverage objective of total coverage*/
if coverage
nottotal_coverage and *notoff_bodyleft* **then** ibcja = ibjca - Δ
off_bodyleft **then** ibcja = ibjca + Δ
*total_coverage*0 **then** ibcja = ibcja
endif
end k_{55}

defrule k_{65} (coverage)
/* Effect of coverage on ibcjb (left splay) with coverage objective of total coverage*/
if coverage
nottotal_coverage and *notoff_bodyright* **then** ibcjb = ibjcb - Δ
off_bodyright **then** ibcjb = ibjcb + Δ
*total_coverage*0 **then** ibcjb = ibcjb
endif
end k_{65}

defrule k_{16} (overlap)
/* Effect of overlap on etamx with overlap objective overlap state 3*/
/* The heuristic used is to always remember the the far field distances */
/* of the last point on the body. When the grid goes offbody then return */
/* it to the last best onbody value.*/
if overlap
1 or 2 **then** etamx = etamx + Δ
3 or 4 **then** etamx = etamx
endif
end k_{16}

defrule k_{26} (overlap, relative_volume)
/* Effect of overlap and relative_volume on deta with overlap objective overlap state 3*/
if overlap
2 **then if** relative_volume
< 1 and (k $> \frac{kmax}{2}$) **then** deta = deta - Δ
< 1 and (k $< \frac{kmax}{2}$) **then** deta = deta + Δ
$= 1$ **then** deta = deta
endif
1 or 3 or 4 **then** deta = deta
endif
end k_{26}

defrule k_{36} (overlap, relative_volume)
/* Effect of overlap and relative_volume on dfar with overlap objective overlap state 3*/
if overlap
2 **then if** relative_volume
< 1 and (k $> \frac{kmax}{2}$) **then** dfar = dfar + Δ
< 1 and (k $< \frac{kmax}{2}$) **then** dfar = dfar - Δ
$= 1$ **then** dfar = dfar
endif
1 or 3 or 4 **then** dfar = dfar
endif
end k_{36}

defrule k_{46} (overlap)
/* Effect of overlap on kmax with overlap objective overlap state 3*/
/* If most farfield points have too little overlap increase kmax to increase overlap*/
if Count ((from j = 1 to jmax) overlap = 2)
$> 50\%$ **then** kmax = kmax + Δ
endif
/* If most farfield points have too much overlap decrease kmax to increase overlap*/
if Count ((from j = 1 to jmax) overlap = 4)
$> 50\%$ **then** kmax = kmax - Δ
endif
end k_{46}

defrule k_{56} (overlap)
/* Effect of overlap on ibcja with overlap objective overlap state 3*/
if overlap
/* splay out*/
1 or 2 **then** ibcja = ibcja - Δ
/* keep splay constant*/
3 **then** ibcja = ibcja
/* splay in*/

```
        4 then ibcja = ibcja + Δ
        endif
end k56

defrule k66 (overlap)
    /* Effect of overlap on ibcjb with overlap objective overlap state 3*/
    if  overlap
        /* splay out*/
        1 or 2 then ibcjb = ibcjb - Δ
        /* keep splay constant*/
        3 then ibcjb = ibcjb
        /* splay in*/
        4 then ibcjb = ibcjb + Δ
        endif
end k66
```

C Users Manual

The grid controller is a program written in Fortran90. It is executed by a command at the shell prompt.

```
% controller controller.inp
%
% more controller.inp
```

```
GRID CONTROLLER INPUT FILE

  1                               GRID SCHEDULE ELEMENT 1
  2,.005                          OBJECTIVES
  20,0,0,1,0                      # of iterations, TYPE OF CONTROL
  2,2,2,1,1,1,1,0                 GRID STATES
  3                               GRID CONTROL LINE SOURCES

  2                               GRID SCHEDULE ELEMENT 2
  2,.005                          OBJECTIVES
  20,1,0,0,0                      # of iterations, TYPE OF CONTROL
  1,1,0,0,0,0,0,0                 GRID STATES
  3                               GRID CONTROL LINE SOURCES
```

C.1 Input File Specification

The input file controller.inp is organized as a grid schedule. At each step of the grid schedule several types of information are specified:

- OBJECTIVES: A set of grid objectives are defined for the active grids. These objectives are in terms of the single and binary grid qualities which the controller must achieve and maintain for each active grid. The user can specify the following objectives: number of cell overlap, maximum truncation error, stretching ratio, and jacobians.
- NUMBER OF ITERATIONS: The user defines the maximum number of feedback iterations the grid controller executes. Keep in mind that the grid controller can terminate earlier if the grid objectives are met or the grid controller reaches a local optima.
- TYPE OF CONTROL: The controller can be executed in one of four modes (per iteration). Modes can be used in conjunction with each other. Each mode is specified as 0: off, or 1: on. The first is overlap control which invokes SURGRD to achieve overlap grid qualities. The second mode repairs regions of low overlap grid qualities with sets of grids. The third mode modifies control lines to achieve truncation error grid qualities. The fourth is the optimizer control mode, (where the grid controller utilizes SRAP) and is used as a subroutine by an optimizer. It requires an additional input file called optimizer.inp.
- GRID STATE: For each grid, a grid state is defined. 0 implies not active, not present, 1 implies active, present and 2 implies not active, present. The grids which will be modified have grid state 1. The grids which will be used to determine grid qualities have grid states 1 and 2.
- GRID CONTROL LINE SOURCES - define initial state source. These sources can be pointers to grids, pointers to control lines with grid specifications or pointers to grid heuristics from which a grid can be generated. There are four values: synthesize grid spec from heuristics (0), use actual grid as starting point due to no grid spec (1), and grid spec from file (2) and use control line from pristine subdirectory (3). The controller will look in subdirectory `$current_dir/pristine_grids/` for the initial grids.

C.2 Outfile Directory Structure

The result of the execution is a subdirectory which contains a log of the grid controller operations as well additional subdirectories for grids, specifications and the results of analysis routines run during the execution of the grid controller. The output of the grid controller is found in the subdirectory `$current_dir/run/`. Under this subdirectory several files and subdirectories can be found.

- `$current_dir/run/gridcontrollerlog`: This file contains the output of the grid controller. Each iteration of the controller is highlighted in the log as well as the reasoning processes the grid controller uses to determine its actions.
- `$current_dir/run/s1`: The results of each grid schedule element are captured in subdirectories labelled s1, s2 ... sn, where the nth grid schedule step is recorded in the sn subdirectory. Each sn contains six subdirectories under it. Each of these subdirectories contains the labelled contents for each iteration run by the grid controller.
- `$current_dir/run/s1/grid_histories_1d`: Contains the control lines for all iterations.
- `$current_dir/run/s1/grid_histories_2d`: Contains the surface grids for all iterations.
- `$current_dir/run/s1/grid_histories_3d`: Contains the volume grids for all iterations required to compute the single and overlap grid qualities.
- `$current_dir/run/s1/sgq_histories`: Contains the single grid qualities for all grids for all iterations.
- `$current_dir/run/s1/olgq_histories`: Contains the overlap grid qualities for all grids for all iterations.
- `$current_dir/run/s1/spec_histories`: Contains the SRAP and SURGRD input specifications synthesized by the grid controller for all grids, control lines for all iterations.